Capítulo 6 LA COMUNICACIÓN ENTRE AGENTES

La comunicación entre los agentes les permite sincronizar acciones, enviar y recibir conocimiento, resolver conflictos en la resolución de una tarea, etc. (Russell *et al.*, 1995). La comunicación es el soporte de la cooperación, coordinación y de los procesos de negociación que se dan entre los agentes.

El proceso de comunicación permite que los agentes desarrollen sus acciones con una visión menos local, tras un proceso de sincronización con el resto de los agentes. Sin embargo, una excesiva comunicación puede dar lugar a agentes cuya sobrecarga de comunicación sea mayor que el trabajo efectivo realizado.

La comunicación permite la coordinación entre un grupo de agentes de forma que se ejecuten solamente aquellas acciones necesarias. Como no es el propósito de esta sección profundizar en las técnicas de coordinación entre agentes nos limitamos a decir que de forma general hay dos modelos de coordinación: (i) Coordinación global: cuando el sistema multiagente es capaz de determinar y planificar globalmente las acciones de los diferentes agentes. En este caso un agente verifica todos los conflictos posibles entre los agentes del sistema. (ii) Coordinación individual: cuando los agentes del

sistema multiagente tienen autonomía total para decidir qué hacer y resolver los conflictos con otros agentes.

Cuando se trata de resolver un problema de manera distribuida, entre un grupo de agentes, la solución es el resultado de la interacción cooperativa entre todos ellos. La comunicación facilita por tanto los procesos de cooperación. El grado de cooperación que hay entre agentes puede ir desde una cooperación completa hasta una antagónica u hostil. En el primer caso se paga un alto coste por la total comunicación entre agentes, pero en el segundo puede ocurrir que unos agentes bloqueen los objetivos de otros.

Para que los mecanismos de cooperación y coordinación tengan éxito en un sistema de agentes, debe de existir un mecanismo adicional, por medio del cual, los integrantes de un sistema se puedan poner de acuerdo cuando cada agente defiende sus propios intereses, llevándolos a una situación que los beneficie a todos, teniendo en cuenta el punto de vista de cada uno. Este mecanismo es la negociación.

6.1. Tipos de Comunicación

Se pude distinguir un amplio rango de formas de comunicación (Werner, 1989):

- No hay comunicación. Los agentes pueden interactuar sin comunicarse, infiriendo las intenciones de otros agentes. Este modelo de comunicación es el que consigue mejores resultados cuando los objetivos de los agentes no están interrelacionados y por lo tanto no pueden entrar en conflicto (Schelling, 1960).

- Comunicación primitiva. En este caso la comunicación está restringida a un número de señales fijas con una interpretación establecida de antemano (Hoare, 1978).

- Arquitectura de pizarra. Se utilizan en el campo de la inteligencia artificial como una manera de compartir memoria y conocimiento (Hayes-Roth, 1985). Los agentes pueden escribir mensajes, dejar resultados parciales o encontrar información en una pizarra que

todos saben donde está. Esta arquitectura se utiliza para intercambiar información entre subsistemas de agentes o para modelar el intercambio entre los distintos módulos que componen la estructura de un agente.

- Intercambio de planes e información. Dos agentes pueden intercambiarse sus respectivos planes, de esta manera cada uno puede adaptar sus estrategias. Esto presenta algunos incovenientes como el alto coste computacional del intercambio.

- Intercambio de mensajes. En este caso los agentes actúan en respuesta al procesamiento de una comunicación (Agha *et al.*, 1988). Las acciones que pueden ejecutar estos agentes son: enviarse mensajes entre ellos mismos, determinar un cambio en el comportamiento como consecuencia de haber alcanzado un nuevo estado después de la comunicación, etc.

- Comunicación de alto nivel. El diálogo entre agentes permite la generación e interpretación de palabras o declaraciones, con el objetivo de comunicar la información que el emisor conoce formada por creencias, compromisos e intenciones, para que así el receptor alcance el mismo estado mental que tiene el emisor (Grosz *et al.*, 1990).

6.2. Actos de comunicación

Antes de comenzar con el estudio de los lenguajes de comunicación resulta interesante analizar qué se entiende por comunicación y qué se quiere representar en un acto de comunicación.

La teoría de la comunicación deriva de los estudios en telecomunicación de Shannon y Weaver en los años 40. En esta teoría, un acto de comunicación se define como el envío de información entre un emisor y un receptor. La información está codificada en un lenguaje conocido tanto por el emisor y como por el receptor, de manera que cuando se recibe se decodificada. La información se envía a través de un medio y en un

determinado contexto (situación en la que se encuentran el emisor y el receptor). El estudio de la comunicación desde este punto de vista está centrado en cuestiones técnicas como: métodos de codificación y decodificación, análisis de los distintos medios y de las perturbaciones que sufren las señales en esos medios, etc. Básicamente el estudio de la teoría de la comunicación se centra en el acto del envío y la recepción (en el intercambio de información) sin estudiar otros aspectos de la comunicación humana como la consciencia de comunicar, la voluntad de comunicar, la comunicación verbal y los distintos modelos de comportamiento como gestos, miradas, la mímica, etc.

Otro problema no abordado es la relación de la comunicación con el proceso de razonamiento interno, es decir, ¿la información comunicada significa lo mismo para los dos agentes implicados? ¿existe una información común compartida por el emisor y el receptor?. Este tipo de cuestiones aparece en el contexto de las ciencias cognitivas y existen propuestas interesantes (Sperber y Wilson, 1986), donde cada agente construye sus propios modelos e intenta transmitir aquello que considera necesario para hacerse comprender. Estos agentes se basan en la utilización de mecanismos inferenciales y en la construcción de modelos (representación de hipótesis y objetivos de otros agentes).

Aún así la teoría de la comunicación es suficiente para describir las características de los actos de comunicación entre los agentes. Además de estas características es necesario estudiar qué desea el emisor del receptor cuando envía un mensaje (la teoría de los actos lingüísticos) y cómo se relacionan estos actos cuando se produce una conversación.

6.2.1. Características de los actos de comunicación

La comunicación entre varios agentes puede describirse por el modo en que ésta se produce. El modo de comunicar se puede analizar desde tres puntos de vista (Ferber, 1999):

- El enlace entre el emisor y el receptor. En este caso la relación puede ser: (i) “punto a punto”, si el agente emisor conoce la dirección del

agente receptor y envía el mensaje directamente al receptor; (ii) "*broadcasting*", en este caso el agente emisor envía el mensaje a todos los agentes que conoce realizando un envío conjunto.

- La naturaleza del medio de comunicación. En este caso la relación puede ser: (i) "directo", el medio recoge el mensaje en el emisor y lo deposita en la dirección del receptor sin considerar la distancia y sin que pueda interceptarse la comunicación por otros agentes (ejemplo de este tipo de medio es el utilizado por el correo electrónico); (ii) "propagación de señales", el medio es un medio físico que el emisor utiliza en este caso la distancia afecta a la comunicación (a mayor distancia más débil es la señal) y además el resto de agentes pueden interceptar la información emitida (ejemplos de este tipo de medio son las comunicaciones entre robots mediante enlaces de radio); (iii) "anuncios públicos", el medio es una especie de tablón de anuncios donde el agente, cuando necesita algo, deposita un mensaje que será recibido por aquellos agentes que accedan al tablón (ejemplos de este tipo de comunicación son las pizarras).

- La intención de la comunicación. En este caso se evalúa si los agentes desean comunicarse o no, es decir si la comunicación es "intencional" o "accidental". La intención del agente emisor está relacionado con el deseo de comunicar con el agente receptor, es decir tan importante es el contenido de la información a comunicar (el qué) como la intención de los agentes (el porqué). Las intenciones se clasifican según (Dennett 1983) en: (i) "tipo 0", cuando el emisor emite una señal debida a su estado interno o a un estímulo captado aunque no exista receptor, se produce una comunicación accidental ya que las emisiones de un agente son captadas por otro (sucede por ejemplo en las comunidades animales entre las presas y los depredadores). El hecho de ser accidental sin que el emisor sea consciente de ello, provoca que no controle el mensaje que envía y al mismo tiempo que el receptor sea accidental implica que la interpretación del mensaje es variable y dependiente del receptor; (ii) "tipo 1", cuando el emisor

envía un mensaje al receptor para que el receptor realice una determinada acción; (iii) “tipo 2”, cuando el emisor envía un mensaje al receptor para que el receptor crea un determinado hecho; (iv) “tipo 3”, cuando el emisor envía un mensaje al receptor para que el receptor crea que otro agente cree un determinado hecho. Evidentemente estos niveles de intencionalidad no son observables directamente desde el exterior, sólo es posible saber el nivel de intencionalidad accediendo al estado interno de los gentes.

6.2.2. Actos lingüísticos

Los actos lingüísticos definen las diversas acciones (que llevan a cabo los agentes de manera intencional) que se desarrollan durante el transcurso de una conversación. Según Searle (1979) y Vanderveken (1992) existen distintos tipos de actos lingüísticos:

- Actos asertivos: estos actos sirven para proporcionar una información del entorno al afirmar un hecho. Ejemplos de estos actos lingüísticos son: “el agente B está colapsado”, “el entorno es rectangular”, etc.

- Actos directivos: estos actos indican al agente que los recibe que debe realizar una determinada acción: Ejemplos de estos actos lingüísticos son: “Dame el resultado de sumar 2 y 2”, “dime de que forma es el entorno”, etc.

- Actos promisivos: estos actos comprometen al emisor del mensaje en la realización de ciertas acciones en el futuro. Ejemplos de estos actos lingüísticos son: “Mañana te envío la información”, “en n ciclos moveré este objeto”, etc.

- Actos expresivos: estos actos permiten al receptor recibir información del estado mental del emisor. Ejemplos de estos actos lingüísticos son: “estoy alegre”, “estoy enfadado”, etc.

- Actos declarativos: estos actos se llevan a cabo por el mero hecho de expresar una declaración. Ejemplos de estos actos lingüísticos son: “se abre la reunión”, “te doy el trabajo”, etc.

Los actos lingüísticos son estructuras complejas compuestas por tres componentes (actos lingüísticos elementales) (Austin, 1962; Searle, 1969):

- La componente locutoria: relacionada con la componente material de la generación de declaraciones. Básicamente está relacionada con el medio material y los mecanismos físicos que permiten generar una declaración a partir de una gramática y de un léxico.

- La componente ilocutoria: relacionada con la ejecución del acto lingüístico. Estos actos se caracterizan por una fuerza (afirmación, pregunta, pedir hacer, prometer, ordenar, informar, etc.) y un contenido. Así por ejemplo: en el acto “Pregunta(está lloviendo)” la fuerza es “Pregunta” y el contenido “está lloviendo”. Esta manera de representar los actos se utiliza en sistemas multiagente ya que este tipo de actos suelen venir definidos por el verbo que se utiliza (pedir, prometer, prguntar, etc.) y que en sistemas multiagente se define como “ejecutiva” como veremos en el apartado de KQML.

- La componente perlocutoria: relacionada con el efecto que causa el acto lingüístico (la componente ilocutoria) en el estado del receptor. Así por ejemplo son actos de este tipo inspirar, convencer, persuadir, luchar, etc. No se definen “ejecutivas” para ellos aunque deben entenderse como consecuencias de las “ejecutiva” de los actos ilocutorios.

Los actos lingüísticos no son verdaderos o falsos más bien se habla de si tienen éxito o fallan. Un acto lingüístico puede fallar debido a los problemas que puedan aparecer en:

- El enunciado del acto: cuando se manda un mensaje este puede tener errores debido al medio o puede ocurrir que la dirección del destinatario no esté bien consignada o que el lenguaje (protocolo) del

receptor sea distinto del lenguaje del emisor por lo que no podrán entenderse.

- La interpretación del acto: puede ocurrir que aunque el acto llegue correctamente al receptor, éste no interprete bien la fuerza de la componente ilocutoria de manera que una posible pregunta sea interpretada como una afirmación y por lo tanto el agente receptor no envíe una respuesta, si no que almacene el contenido de la pregunta en su estado interno como un hecho cierto.

- La falta de capacidad para realizar dicho acto: en este caso aunque se recibe correctamente el acto lingüístico y se interpreta de forma adecuada el agente receptor no cuenta entre sus capacidades con aquella que necesita en ese momento. En general podemos distinguir cuando claramente el agente receptor no puede (por ejemplo por no tener ninguna habilidad para realizar una tarea) de aquellas situaciones en las que no se tiene una certeza absoluta de que no pueda (promesas, actos futuros) y por lo tanto se debe manejar un cierto riesgo con respecto a si será capaz o no de realizar dicho acto.

6.2.3. Modelado de las Conversaciones

La teoría de los actos lingüísticos vista en el apartado anterior establece las características de cada acto de forma individual. Cuando se establece una conversación se produce un intercambio de actos entre dos agentes que da lugar a un conjunto de acciones y de cambios de estado interno en función de los distintos actos.

Para cada acto recibido por un agente deben definirse el conjunto de actos posibles que puede realizar y que en algunos casos darán lugar a un acto de respuesta. Evidentemente este conjunto de posibles respuestas a un acto determinado deben definirse para cada uno de los agentes y definirá el comportamiento social del mismo.

Para modelar este conjunto de posibilidades se pueden utilizar autómatas finitos o redes de Petri. El modelo basado en autómatas finitos (Winograd y

Flores, 1986) se utiliza para la representación de la comunicación de un agente involucrado en una única conversación. Así cada uno de los estados internos del agente se representa mediante un estado en el autómata finito y los actos lingüísticos que recibe y que emite le hacen transitar entre los estados. El modelo basado en redes de Petri se utiliza para representar múltiples conversaciones al mismo tiempo (Estraillier y Girault, 1992) aunque en el contexto de los sistemas multiagente todavía no ha tenido una amplia aceptación.

6.3. Lenguajes de comunicación de agentes

El hecho de que los agentes se comuniquen tanto con otros agentes como con el medio en el que habitan es un elemento representativo de este tipo de sistemas. Además es fundamental que exista un mecanismo adecuado de comunicación entre agentes sobre todo en comunidades multiagente. Incluso se ha llegado a sugerir que una entidad es un agente software si y sólo si se comunica correctamente en un lenguaje de comunicación de agentes (Genesereth *et al*., 1994).

Para que esta interacción e interoperación entre agentes software sea significativa, constructiva e inteligente se requieren tres componentes fundamentales y distintos:

- Un protocolo de transporte: formado por el mecanismo de transporte usado para la comunicación (por ejemplo: TCP, SMTP, HTTP, etc).

- Un lenguaje común: es el medio por el cual se realiza el intercambio de información. Este lenguaje indica cual es el contenido de la comunicación y si es una aseveración, una pregunta o una solicitud.

- Un protocolo de interacción: se refiere a la estrategia que sigue el agente para interactuar con otros agentes, la cual puede ir desde esquemas de negociación y protocolos basados en teoría de

juegos hasta protocolos muy simples en los que cada vez que el agente no sabe algo busca otro que lo sepa y le pregunta.

Los lenguajes de comunicación comunes facilitan la creación de software interoperable. De forma que los agentes o componentes pueden comunicarse independientemente del lenguaje en el que han sido implementados. En este caso se establece una separación entre la implementación de los agentes o componentes y del interface. Existen actualmente dos puntos de vista en el diseño de estos lenguajes:

- Lenguajes procedurales: la comunicación se modela como un intercambio de directivas procedurales, pudiendo transmitir además de comandos, programas enteros que permiten a los agentes alcanzar sus objetivos. Las ventajas de estos lenguajes procedurales es que son sencillos y su ejecución es eficiente y directa. Sin embargo, y desafortunadamente, existen desventajas. Estos lenguajes requieren que se tenga información sobre los agentes que recibirán los mensajes, información que a veces no está disponible para el agente emisor. También existe el problema de que los procedimientos son unidireccionales. Mucha de la información que los agentes requieren compartir debe estar disponible en ambas direcciones (transmisión-recepción, y viceversa). Lo anterior puede complicarse cuando un agente recibe varios “scripts” provenientes de múltiples agentes que trabajan simultáneamente y que pueden interferir entre sí. La mezcla de información procedural es más complicada que la de información de tipo declarativa. Suelen usarse lenguajes de intérpretes de órdenes (scripts) tales como Perl, Tcl (Gray *et al.*, 1997), Apple Events, Telescript (White, 1994) etc. estos languajes permiten un rápido prototipado aunque no suelen ser fácilmente escalables ni reciclables. Son especialmente útiles para la construcción de agentes en aplicaciones finales como por ejemplo agentes de usuario o agentes móviles.

- Lenguajes declarativos: la comunicación tiene lugar utilizando enunciados declarativos, es decir intercambio de declaraciones (definiciones, suposiciones, etc), por lo que necesita que el lenguaje sea lo suficientemente expresivo (como para comunicar diferentes clases de información) y compacto. Varios Proyectos ARPA han hecho posible que la iniciativa Knowledge Sharing Effort haya permitido definir los componentes del lenguaje de comunicación de agentes KQML (Knowledge Query Manipulation Language) (Genesereth *et al*., 1994) (ARPA, 1992). El otro estándar ha sido desarrollado por FIPA (Foundation for Intelligent Physical Agent) y se denomina ACL (Agent Communication Language). Al final las diferencias entre uno y otro lenguaje han desaparecido y se puede hablar de un único estándar donde el lenguaje ACL es una generalización del KQML como veremos en el próximo apartado.

6.4. ACL (Agent Communication Language)

A continuación nos centramos en la definición del lenguaje ACL (Agent Communication Language) por ser uno de los más utilizados y porque a nuestro juicio muestra la esencia de este tipo de lenguajes. ACL es un lenguaje que permite la interoperación entre agentes autónomos distribuidos. Un mensaje en ACL es una expresión KQML qué consiste en una directiva de comunicación y un contenido semántico en KIF (Knowledge Interchange Format) (es decir los argumentos de la expresión son términos u oraciones expresados en KIF) construido a partir de términos de un vocabulario. En definitiva, ACL tiene tres componentes:

- Vocabulario.
- KIF (Knowledge Interchange Format).
- KQML (Knowledge Query Manipulation Language).

6.4.1. Vocabulario. Ontologías

El vocabulario del ACL es un diccionario abierto de palabras amoldadas a cada una de las áreas del dominio de la aplicación. Cada palabra tiene una descripción en lenguaje natural (que facilita el entendimiento de su significado) y una anotación formal (escrita en KIF). El diccionario está abierto de forma que se puedan añadir palabras de nuevas áreas de aplicación.

El término ontología se utiliza para definir la especificación de una conceptualización (Gruber, 1993), en este caso permite hacer una descripción de los conceptos y relaciones que pueden formar parte del conocimiento de un agente o de una comunidad de agentes. Las ontologías utilizan un vocabulario formal, y un conjunto de definiciones.

Los agentes establecen compromisos ontológicos. Un compromiso ontológico es un acuerdo para usar un vocabulario, que es consistente, aunque no completo, respecto a la teoría especificada en la ontología, es decir; se trata de definir un vocabulario común con el que se puede representar el conocimiento compartido. Se construyen agentes comprometidos con ontologías y se diseñan ontologías con las que los agentes pueden compartir conocimiento.

Un área de aplicación se puede describir de muchas formas, el diccionario puede contener múltiples ontologías para un área y cada agente utiliza la que más le conviene. Por ejemplo puede definirse un vocabulario para describir la geometría tridimensional en términos de coordenadas rectangulares, coordenadas polares, coordenadas cilíndricas, etc. Los agentes pueden utilizar las definiciones formales asociadas a las ontologías para transformar mensajes de una ontología a otra ontología.

En las siguientes direcciones electrónicas se puede obtener más información sobre ontologías: http://AI-WWW.AIST-NARA.AC.jp/~takeda/doc/html/icot-paper-v2/, http://www.ksl.stanford.edu/knowledge-sharing/ontologies/README.html.

6.4.2. KIF *(Knowledge Interchange Format)*

El lenguaje de contenido KIF (Knowledge Interchange Format, Formato de intercambio de conocimiento o lenguaje interno) es una versión en notación prefija (es decir, el símbolo que define la operación a realizar se antepone a los operandos) del cálculo de predicados de primer orden, con varias extensiones para incrementar su expresividad. La descripción del lenguaje incluye tanto una especificación para su sintaxis como una para su semántica (Genesereth *et al.*,1992).

KIF permite expresar datos simples e información más complicada (restricciones, negaciones, disyunciones, reglas, expresiones, e información a un metanivel) mediante el uso de términos complejos. Además, KIF incluye una variedad de operadores lógicos para ayudar a codificar información lógica y dos operadores (? y ,) junto a un vocabulario que permita codificar conocimiento acerca del conocimiento. Por último, también se puede utilizar para describir procedimientos, es decir, escribir programas para agentes. Desde el punto de vista sintáctico KIF es similar a Lisp y a Scheme.

En definitiva, KIF define un conjunto de objetos, funciones y relaciones cuyo significado es fijo, aunque KIF también es abierto, es decir, los usuarios tienen la libertad de definir significados de cualquier otro símbolo que no esté predefinido. En la dirección electrónica http://www.cs.umbc.edu/kse/kif/ está disponible información sobre KIF y herramientas asociadas.

6.4.3. KQML (Knowledge Query Manipulation Language)

Aunque es posible diseñar un marco de trabajo completo para la comunicación en el que todos los mensajes tengan la forma de oraciones en KIF, esto sería ineficiente, ya que se requeriría incluir información implícita sobre el agente que envía el mensaje y sobre el que lo recibe, junto con información complementaria como por ejemplo la hora del envío del mensaje, la historia del mensaje, etc., debido a que la semántica de KIF es independiente del contexto. La comunicación es más eficiente si se provee un elemento lingüístico en la que el contexto se toma en cuenta. Esta es la

función de KQML (Knowledge Query Manipulation Language) (Mayfield *et al*., 1995).

El lenguaje de comunicación KQML (Lenguaje de manipulación y consulta de conocimiento) define un conjunto de protocolos que facilitan el intercambio de información y conocimiento entre agentes (programas autónomos y asíncronos), en tiempo de ejecución. Este lenguaje esta basado en la teoría de actos del habla, que se utiliza para construir una capa lingüística en la que se tiene en cuenta el contexto y para formalizar las acciones lingüísticas de los agentes. Esta teoría se basa en el hecho de que las oraciones expresadas por humanos durante la comunicación no siempre aseveran un hecho, sino que en realidad tratan de transmitir una creencia o conocimiento, una intención o un deseo. Además, esta teoría también ha contribuido al entendimiento de la relación entre el estado interno de un agente y las expresiones que intercambia con otros agentes (Haddadi, 1996).

KQML define el formato de los mensajes y el protocolo que los maneja para permitir que los agentes se identifiquen, se conecten e intercambien información con otros agentes. Sus características son (Mayfield *et al.*, 1996):

KQML cumple con los estándares definidos por FIPA (Foundation for Intelligent Phisical Agents, 1997).

- Los mensajes son opacos al contenido de lo que transportan, es decir, no comunican únicamente oraciones en un lenguaje, sino que comunican una actitud acerca del contenido. Por ejemplo, afirmación, solicitud, pregunta.

- Las primitivas del lenguaje se llaman ejecutivas (performative, actuación o realizativos), e indican qué acciones u operaciones pueden llevar a cabo los agentes cuando se comunican, es decir; las operaciones que los agentes pueden ejecutar en la base de conocimiento de los otros agentes.

- Para la comunicación entre agentes, se pueden usar agentes especiales llamados facilitadores (*communication facilitators*) que coordinan las relaciones e interacciones entre los agentes, es decir, proporcionan ciertas funciones tales como: asociación de direcciones físicas con nombres simbólicos, registro de bases de datos y/o servicios ofrecidos o buscados por los agentes; y servicios de comunicación como reenvío e intermediación (*brokering*).

Para poder identificar cada uno de los componentes de ACL, utilizaremos como ejemplo un mensaje en el que aparecen los tres componentes.

Mensaje: Petición de información sobre el trabajador ideal para llevar a cabo una taréa concreta.

ask-if

:content (<*tarea-concreta*>)

:reply-with q_j

En negrita aparece la ejecutiva KQML y sus parametros, en cursiva el vocabulario (representan objetos del dominio de la aplicación) y el resto (q_j, que es el identificador del mensaje, los parentesis y demás) es KIF.

6.5. Descripción de KQML

El lenguaje KQML conceptualmente está estructurado en tres niveles (Finin *et al.*, 1994):

- Nivel de contenido: se relaciona con el contenido real del mensaje escrito en el lenguaje de representación propio de cada agente (se puede usar cualquier lenguaje incluyendo lenguajes expresados como cadenas en ASCII o aquellos expresados usando una notación binaria). Cualquier implementación de KQML ignora la

parte del contenido excepto la magnitud que se utiliza para determinar donde termina.

- Nivel de mensaje: es el núcleo de este lenguaje pues se utiliza para codificar el mensaje que un agente desea transmitir a otro, es decir; determina las clases de interacciones que se pueden tener con un agente que “hable” KQML. La función principal de este nivel es identificar el protocolo que se va a usar para entregar el mensaje (síncrono o asíncrono) y proveer un acto del habla o ejecutiva que el transmisor agrega al contenido. La ejecutiva indica si el contenido es una aserción, una pregunta, un comando, etc. Además, ya que el contenido es opaco a KQML, en esta capa se incluyen características opcionales que describen el contenido: su lenguaje, la ontología que se usa y alguna clase de descripción más general, como por ejemplo un descriptor que identifica un tema dentro de la ontología. Estas características permiten que las implementaciones de KQML analicen, ruteen y entreguen el mensaje apropiadamente aunque su contenido sea inaccesible.

- Nivel de comunicación: codifica un conjunto de características del mensaje que describen parámetros de comunicación de bajo nivel, tales como la identidad del agente que envía el mensaje y del que lo recibe y un identificador único asociado con la comunicación.

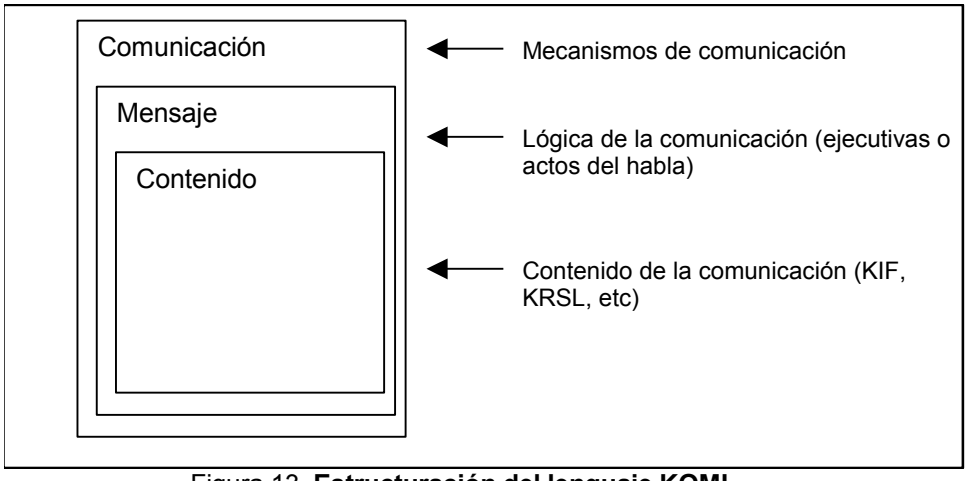

Figura 13. **Estructuración del lenguaje KQML.**

6.5.1. Sintaxis KQML

El lenguaje KQML es independiente del formato de la información. Un mensaje KQML se inicia con una ejecutiva (indica la intención del mensaje o el acto del habla, y se representa como una cadena ASCII, además invita al receptor a realizar alguna acción), seguida de sus argumentos asociados que incluyen el contenido real del mensaje, y un conjunto de argumentos opcionales. Estos argumentos describen el contenido de una manera independiente de la sintaxis del lenguaje utilizado para expresar el contenido.

La sintáxis de KQML se ayuda de una lista de paréntesis balanceados y se puede considerar como una subclase de la notación prefija del CommonLisp. El elemento inicial de la lista es la ejecutiva y el resto de elementos son los argumentos, que son pares de palabra-clave/valor. Los parámetros están indexados por palabras clave, que pueden usarse en cualquier orden. Las palabras clave, llamadas *parameter names,* deben empezar por dos puntos (:), y deben ir delante de sus correspondientes valores, llamados en la especificación *parameter values*. Debido a que el lenguaje es relativamente simple, la sintáxis actual es relativamente insignificante y se puede cambiar si es necesario en el futuro.

En la siguiente tabla se puede observar la sintaxis de KQML en formato BNF (BACKUS-NAUR FORM). Cuando se utiliza esta sintáxis es conveniente tener en cuenta las siguientes indicaciones:

[<x>] significa “opcional <x>” <x>* significa “cero o más <x>”

<x>+ significa “uno o más <x>” <x>|<y> significa “<x> or_exclusivo <y>”

```
<performative> ::=(<word> {<whitespace> :<word> <whitespace> <expression>}*)

<expression> ::= <word> | <quotation> | <string> | (<word> {<whitespace> <expression>}*)

<word> ::= <character> <character> *

<character> ::= <alphabetic> | <numeric> | <special>

<special> ::= < | > | = | + | - | * | / | & | ^ | ~ | _ |  @ | $ | % | : | . | ! | ?

<quotation> ::= '<expression> | ,<comma-expression>

<comma-expression> ::= <word> | <quotation> | <string> | ,<comma-expression>  (word> {<whitespace>
<comma-expression>}*)

<string> ::= "<stringchar>*" | #<digit><digit>*"<ascii>*

        <stringchar> ::= \<ascii> | <ascii>-\-<double-quote>
```

Tabla 4. **Sintaxis BNF de la gramática de KQML.**

Los nombres de las ejecutivas son palabras reservadas, y existe un conjunto de ellas estándar que se dividen en:

- ejecutivas de discurso: se emplean en el contexto de un intercambio de información y conocimiento entre dos agentes.

  ask-if, ask-all, ask-one, tell, deny, achieve, advertise, subscribe, etc.

- ejecutivas de intervención y mecánicas de la conversación: se usan para intervenir en el curso normal de una conversación.

  error, sorry, ready, next, discard, rest, standby, etc.

- ejecutivas de red y de facilitación: no son actos del habla, pero permiten a los agentes encontrar otros agentes capaces de procesar sus mensajes.

  register, forward, broadcast, recommend-one, recruit-all, etc.

Las ejecutivas definidas pueden no ser suficientes para implementar un sistema multiagente, o puede que estos no necesiten soportar todo el conjunto de ejecutivas. Es decir, los agentes utilizaran un subconjunto y

podrán utilizar ejecutivas que no aparecen en la especificación del lenguaje, definiendolas previamente de modo preciso y con el mismo estilo que las de la especificación. Es decir, el conjunto de ejecutivas de KQML es extensible.

6.5.2. Semántica KQML

El modelo semántico de KQML define un contexto simple y uniforme para que los agentes tengan conocimiento acerca de las capacidades de otros agentes (Labrou y Finin, 1994, 1997). Desde fuera, cada agente, aparece como si manejara una base de conocimiento (Knowledge Base, KB). La implementación de un agente no siempre requiere la existencia de una base de conocimiento, sino que puede utilizar un sistema simple de base de datos o una estructura de datos propia. Por ello, se dice que cada agente organiza una base de conocimiento virtual (Virtual Knowledge Base, VKB). Las VKB tiene dos componentes separados:

- Creencias (almacén de información): codifican información sobre el agente y sobre su entorno, incluyendo las VKB de otros agentes.
- Objetivos o intenciones (almacén de metas): codifican los estados del entorno que el agente quiere alcanzar cuando actúa.

Los agentes se intercambian el contenido de sus VKB y el contenido de las de los otros; pero la codificación del contenido de las distintas VKB puede usar gran variedad de lenguajes de representación.

Las ejecutivas primitivas están definidas en términos de su efecto sobre estos almacenes. Por ejemplo, un *tell(s)* es una aserción que hace el emisor al agente receptor para comunicarle que la sentencia *s* está en su almacén de creencias virtual. Un *achieve(s)* es una petición del emisor al receptor para añadir *s* a su almacén de intenciones.

Las ejecutivas tienen parámetros que se identifican por palabras claves. Todos los mensajes que usen parámetros con estas palabras claves deben ser consistentes con su definición. La siguiente tabla muestra el significado de los parámetros más comunes:

Palabra clave	Significado
:content	Contenido de la ejecutiva
:force	El emisor nunca negará el significado de una ejecutiva
:in-reply-to	La etiqueta esperada en una respuesta
:language	El nombre del lenguaje de representación del parámetro *:content*
:ontology	El nombre de la ontología usada en el parámetro *:content*
:receiver	El receptor del mensaje
:reply-with	El receptor espera una respuesta con etiqueta
:sender	El emisor del mensaje

Tabla 5. **Parámetros de las ejecutivas de KQML.**

6.5.3. Requisitos de Transporte

La especificación de KQML (Finin *et al.,* 1993) no trata de estandarizar cómo ha de ser la infraestructura de transporte de mensajes, ya que normalmente esto depende del lenguaje de programación y de la red. Lo que sí define es una abstracción a nivel de transporte:

- Los agentes están conectados mediante links de comunicación unidireccionales.
- Los links pueden tener un pequeño retraso asociado.
- Cuando un agente recibe un mensaje, sabe desde qué link de entrada ha llegado el mensaje.
- Cuando un agente envía un mensaje puede dirigirlo a un link de salida concreto.
- Los mensajes dirigidos a un destino único han de llegar en el orden en que fueron enviados.
- La entrega de mensajes es segura y fiable.

Esta abstracción se puede implementar de diversas formas. Por ejemplo, los links podrían ser conexiones TCP/IP sobre Internet, que están activos solamente mientras dura la transmisión de un mensaje. Los links podrían ser caminos fijos de correo electrónico. También pueden ser conexiones IPC entre distintos procesos ejecutados en máquinas UNIX. KQML considera que a nivel de agente, la comunicación es un paso de mensajes punto a punto.

6.5.4. Protocolos del KQML

Existe una variedad de protocolos de intercambio de información entre procesos, pero en orden de complejidad los que soporta KQML son (Finin *et al.,* 1994):

- Comunicación síncrona (Figura 14): una pregunta bloqueante espera por una respuesta esperada. Un proceso (cliente) envia una pregunta a otro proceso (servidor) y espera por una respuesta. Esto ocurre comúnmente cuanto un razonador encadenado-hacia-atrás recupera información de un origen remoto. Cuando necesita datos, coloca preguntas y espera por las contestaciones antes de intentar cualquier inferencia.

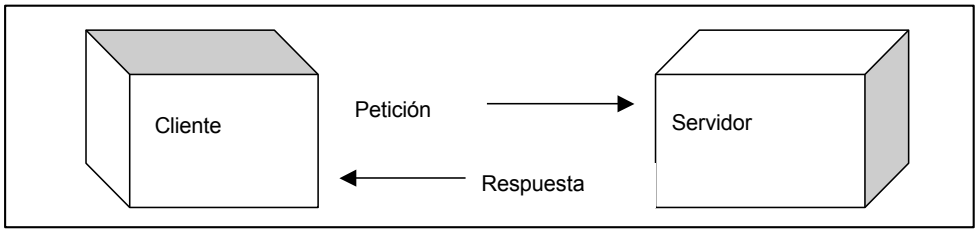

Figura 14. **Comunicación síncrona: una pregunta bloqueante espera por una respuesta esperada.**

Cuando la contestación del servidor(es) está fraccionada, el cliente se ve en la necesidad de encuestar al servidor para obtener la respuesta completa. Un ejemplo de este tipo de intercambio estaría representado por un cliente que interroga una base de datos relacional o un razonador que puede producir una secuencia de instanciaciones en respuesta a una pregunta. Aunque este intercambio requiere que el servidor mantenga

algún estado interno, las transacciones individuales se tratan igual que si se tratara de una sola contestación, es decir; cada transacción es un intercambio “enviar-una-pregunta / esperar / recibir-una-contestación” (Figura 15). Estas transacciones son como las síncronas porque los mensajes llegan al cliente sólo cuando son esperadas.

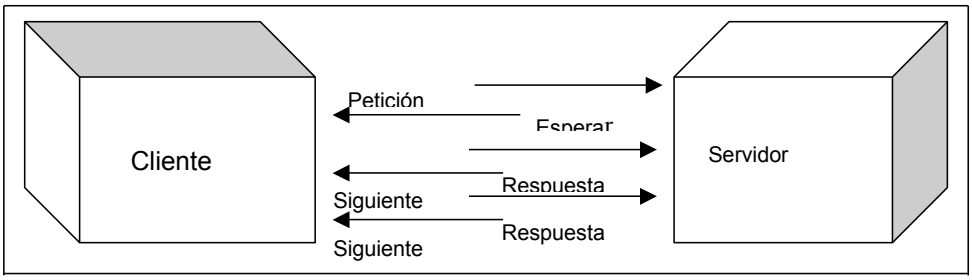

Figura 15. **Comunicación síncrona: las respuestas se envían individualmente a cada petición del cliente.**

- Comunicación asíncrona (Figura 16): Los sistemas de tiempo real trabajan de forma diferente, en ellos el cliente se suscribe a la salida de un servidor(es) y para que así le lleguen un número indefinido de contestaciones a intervalos regulares. En este caso, el cliente no conoce cuando llegará cada mensaje de contestación y puede estar ocupado haciendo cualquier otra tarea cuando lleguen.

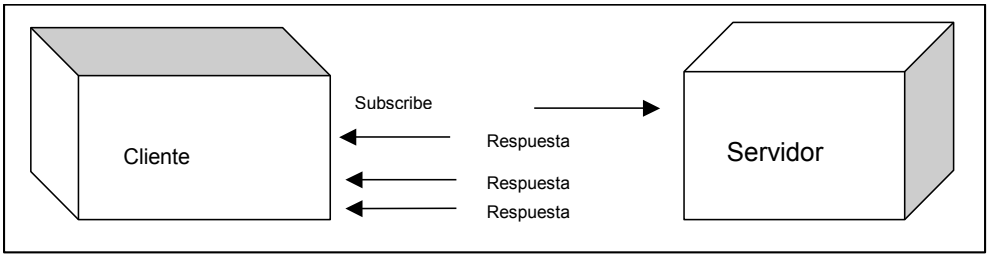

Figura 16. **Comunicación asíncrona.**

Existen más variaciones de estos protocolos. Por ejemplo, los mensajes podrían no estar dirigidos a hosts específicos, sino transmitidos a un número de ellos. Las contestaciones, llegando síncronamente o asíncronamente, se tienen que localizar y, opcionalmente, asociar con la pregunta que contestan.

6.5.5. Arquitecturas del KQML

Debido a que KQML fue diseñado por un comité de representantes de diferentes proyectos (todos preocupados por la manipulación de una colección de procesos que cooperan y por la simplificación de los requerimientos de programación para implementar un sistema de este tipo) no se ha impuesto una arquitectura particular para el sistema. Más concretamente, uno de los criterios en el diseño del KQML fue producir un lenguaje que pudiese soportar una gran variedad de arquitecturas de agentes.

Finin *et al.* (1994) presentan una arquitectura de comunicación construida alrededor de dos programas especializados: un *router* y un *facilitador*, además de una *librería de rutinas de interface* llamada KRIL. La Figura 17 ilustra esta arquitectura de comunicación.

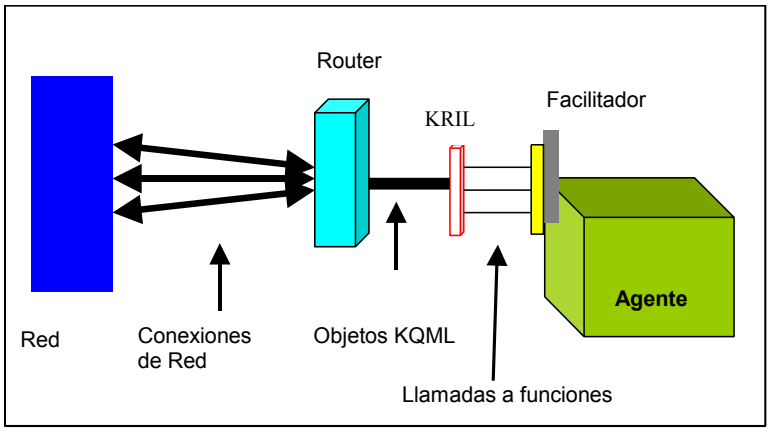

Figura 17. **Arquitectura de comunicación.**

A continuación se describen cada una de los elementos de dicha arquitectura.

- Routers.

  Cada agente software que "habla" KQML está asociado con su propio proceso router independiente. Todos los routers son idénticos siendo todos ellos una copia de uno dado. Un router entrega (encamina) todos los mensajes KQML que van a y desde su agente asociado. Debido a que cada programa tiene un proceso router asociado, no es necesario

hacer extensos cambios en la organización interna del programa(s) para permitirle recibir mensajes de forma asíncrona desde una variedad de orígenes independientes.

El *router* proporciona este servicio al agente y le suministra un solo punto de contacto con el resto de la red. Proporciona tanto al cliente como al servidor funciones para la aplicación y puede manejar múltiples conexiones simultaneas con otros agentes.

El *router* nunca examina los campos del contenido del mensaje que encamina. Cuenta solamente con las ejecutivas de KQML y sus argumentos. Si un mensaje KQML especifica una dirección particular de Internet, el *router* dirige el mensaje a ella. Si el mensaje especifica un servicio particular por el nombre, el *router* intentará encontrar una dirección de Internet para ese servicio y le entrega el mensaje. Si el mensaje sólo proporciona una descripción del contenido (por ejemplo la pregunta, : ontology “geo-domain-3”, :language “Prolog”, etc) el *router* debe intentar encontrar un servidor que pueda ocuparse del mensaje y lo entregará allí, o puede remitirlo a un agente de comunicación más “inteligente” que puede hacerse cargo de él. Los *routers* pueden ser implementados con diversos grados de sofisticación.

Finin *et al.* (1994) han usado esta arquitectura para realizar dos implementaciones del KQML: una en Common Lisp y la otra en C. Ambas son completamente interoperables y frecuentemente se usan juntas. En la implementación hecha en C, un *router* es un proceso independiente de Unix que lanza la aplicación (el agente) y por lo tanto hijo de esta. El canal de comunicación entre el *router* y la aplicación transporta mensajes KQML. Como hay un canal privado entre el *router* y la aplicación no tiene que atenerse al protocolo de KQML y puede transportar más mensajes de lo que está especificado por el protocolo formal. El *router* sólo tiene que tener en cuenta las reglas formales del KQML cuando habla al entorno. La implementación Lisp usa las primitivas de multitarea de Lucid para implementar el *router* como una tarea independiente de Lisp.

- Facilitadores (Finin *et al.,* 1995).

  Para entregar mensajes que tienen la dirección incompleta, los routers cuentan con los facilitadores que son aplicaciones que proporcionan servicios de red útiles. El servicio más simple que proporcionan es mantener un registro con los nombres de los servidores. Los facilitadores ayudan a los routers a encontrar los hosts. Los facilitadores son por tanto consultores en el proceso de comunicación.

  Los facilitadores son realmente agentes software de red; tienen sus propios routers KQML para mantener su tráfico y tratan exclusivamente mensajes en KQML. Típicamente existe un facilitador para cada grupo local de agentes, aunque puede haber múltiples facilitadores locales para proporcionar mayor seguridad. La base de datos del facilitador puede estar implementada en cualquier formato dependiendo del número de servidores hosts y de la calidad de servicio requerida. En sus comienzos la implementación de un facilitador reproducía la base de datos en cada máquina de la red local, para reducir la sobrecarga de la comunicación. Aunque esto ha sido sustituido por una implementación más centralizada para simplificar su implementación, en redes más grandes, en los que es necesario que los facilitadores sirvan múltiples redes, una implementación distribuida (análoga al servidor de nombres del dominio de Internet) puede ser más apropiada.

  Cuando cada aplicación/agente se pone en marcha, su router se anuncia al facilitador local para que sea registrado en la base de datos local. De esta forma las aplicaciones se pueden encontrar unas a otras sin tener que mantener manualmente la lista de servicios locales.

- KRILs.

  Como el router es un proceso independiente de la aplicación, es necesario tener una interface de programación entre la aplicación y el router. Esta interface se llama KRIL (KQML Router Interface Library). Mientras el router es un proceso independiente, que no entiende los campos contenidos en el mensaje KQML, el KRIL está empotrado en la

aplicación/agente y tiene acceso a las herramientas de la aplicación(es) para analizar su contenido. La meta general del KRIL es hacer accesos al router tan simples como sea posible para el programador.

Con este fin, un KRIL puede estar empotrado firmemente en la aplicación, o incluso en el lenguaje de programación de aplicación(es), como es deseable. Sin embargo, no es necesario que esté completamente incluido en el lenguaje de programación de aplicación(es). Un simple KRIL para un lenguaje, generalmente proporciona dos entradas programáticas. Para iniciar una transacción hay una función "send-kqml-message". Ésta acepta el contenido de un mensaje y tanta información sobre el mensaje y su destino como pueda ser proporcionada. Devuelve una contestación al agente(s) remoto(s) (si la transmisión del mensaje es síncrono y el proceso se bloquea hasta que recibe una respuesta) o un simple código indicando por ejemplo que el mensaje se ha recibido.

Para manejar la entrada de mensajes asíncronos, se utiliza normalmente la función declare-message-handler. Esta permite al programador de la aplicación declarar que funciones se pueden invocar cuando lleguen los mensajes. Dependiendo de las capacidades del KRIL, los mensajes de entrada se pueden ordenar de acuerdo a la ejecutiva, al tema o a otras características.

6.6. Ejecutivas del lenguaje KQML¡Error! Marcador no definido.

La tabla siguiente presenta una breve descripción de las ejecutivas del lenguaje KQML.

Nombre	Significado
Achieve	S quiere que R haga algo verdad en su entorno
Advertise	S es particularmente adecuado para procesar una ejecutiva
ask-about	S quiere todas las sentencias relevantes de la VKB de R

Nombre	Significado
ask-all	S quiere todas las respuestas de R a una pregunta
ask-if	S quiere conocer si la sentencia esta en la VKB de R
ask-one	S quiere una de las respuestas de R a una pregunta
Break	S quiere que R rompa el pipe establecido
Broadcast	S quiere que R envie una ejecutiva por todas las conecsiones
Broker-all	S quiere que R coleccione todas las respuestas a una ejecutiva
Broker-one	S quiere conseguir ayuda respondiendo a una ejecutiva
Deny	La ejecutiva incluida no se aplica a S (nada más)
Delete	S quiere que R quite una sentencia de su VKB
Delete-all	S quiere que R quite todas las sentencias emparejadas de su VKB
Delete-one	S quiere que R quite una sentencia emparejada de su VKB
Discard	S no querra las restantes contestaciones de R a una ejecutiva previa
Eso	Fin de un flujo de respuestas a una pregunta prematura
Error	S considera que el mensaje prematuro de R esta mal formado
Evaluate	S quiere que R simplifique la sentencia
Forward	S quiere que R encamine una ejecutiva
Generator	El mismo que una lista de un flujo-completo
Insert	S pide a R que añada el contenido a su VKB
Monitor	S quiere actualizar la respuesta de S a un flujo-completo
Next	S quiere la respuesta de R a una ejecutiva previamente mencionada
Pipe	S quiere dirigir todas las ejecutivas adicionales a otro agente
Ready	S esta listo para responder a la ejecutiva previamente mencionada de R

Nombre	Significado
Recommend-all	S quiere todos los nombres de los agentes que puedan responder a una ejecutiva
recommend-one	S quiere el nombre de un agente que pueda responder a una ejecutiva
recruit-all	S quiere que R consiga todos los agentes convenientes para responder a una ejecutiva
recruit-one	S quiere que R consiga otro agente para responder a una ejecutiva
register	S puede repartir ejecutivas a algún agente nombrado
reply	Comunica una contestación esperada
rest	S quiere las respuestas restantes de R a una ejecutiva previamente mencionada
sorry	S no puede proporcionar una contestación más informativa
standby	S quiere que R este listo para responder a una ejecutiva
stream-about	Versión multiple-respuesta de ask-about
stream-all	Versión multiple-respuesta de ask-all
subscribe	S quiere actualizar la respuesta de R a una ejecutiva
tell	La sentencia en la VKB de S
transport-adress	S asocia el nombre simbólico con la dirección de transporte
unregister	Negación de un registro
untell	La sentencia no está en la VKB de S

6.7. Resumen y Conclusiones

En este capítulo se ha estudiado el proceso de comunicación entre agentes. Por una parte se han visto los distintos tipos de comunicación y cómo clasificarlos y por otra la importancia de la intención en el proceso de

comunicación. Para explicar la intención en la comunicación se ha presentado la teoría de actos lingüísticos de una manera muy somera pero que permite al lector darse cuenta de la importancia y los problemas que conlleva la introducción de la intención en el proceso de comunicación.

Por último se ha presentado una descripción muy detallada del estándar de comunicación ACL-KQML que representa la convergencia de dos estándares y que permite al lector entender la problemática de la comunicación a distintos niveles: la sintaxis del protocolo, la semántica y la intención.

6.8. Referencias

Agha G. y Hewitt C (1988). *Concurrent programming using actors. Object-Oriented Concurrent Programming*. MIT Press.

ARPA *Knowledge Sharing Initiative. Specification of the KQML agent-communication language*. ARPA Knowledge Sharing Initiative, External Interfaces Working Group working paper. Diciembre 1992.

Austin J. L. (1962) *How to do things with words*. Clarendon Press.

Dennet D.C. (1983) "Intentional Systems in Cognitive Ethology: the Panglossian Paradigm Defended". *The Behavioural and Brain Sciences*. Nº 6, pp. 343-390.

Estraillet P., Girault C. (1992) "Applying Petri Net Theory to the Modelling Analysis and Prototyping of Distributed Systems". *Proceeding of the IEEE International Workshop on Emerging Technologies and Factory Automation*. Cairns, Australia.

Ferber, J., (1999) *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*, Addison-Wesley.

Finin T., Labrou Y. y Mayfield J. (1995) "KQML as an agent communication language". *Reporte técnico. Departamento de Ciencias Computacionales. Universidad de Maryland Baltimore County*. Baltimore, MD. 1995.

Finin T., McKay D., Fritson R., McGuire y McEntire R. (1994) "KQML: An information and knowledge exchange protocol". En *International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, December 1993. Una versión de este documento aparece en Kazuhiro Fuchi y Toshio Yokoi (Eds.), Knowledge Building and Knowledge Sharing, Ohmsha y IOS Press, 1994.

Finin T., Weber J., Wiederhold G., Genesereth M., Fritson R., McGuire J., Shapiro S. y Beck C. (1993) "Specification of the KQML Agent-Communication Language". *The DARPA Knowledge Sharing Initiative External Interfaces Working Group*. Junio 1993.

Foundation for Intelligent Physical Agents (1997) *FIPA 97 Specification*, Version 2.0. Part2: Agent Communication Language. Ginebra, Suiza. Octubre 1997.

Genesereth M. R. y Fikes R. (1992) "Knowledge Interchange Format, version 3.0 reference manual". *Technical report, Computer Science Department, Stanford*

University, Junio 1992.

Genesereth M. R., Singh N. P. y Syed M. A. (1994) "A distributed and anonymous knowledge sharing approach to software interoperation". *Reporte técnico. Departamento de Ciencias Computacionales. Universidad de Standford*. Menlo Park, California. 1994.

Gray R. S., Cybenko G., Kotz D. y Rus D. (1997) "Agent Tcl". In W. Cockayne and M.Zypa, editors, *Itinerant Agents: Explanations and Examples with CDROM*. Manning Publishing, 1997.

Grosz B. y Sidner C. (1990) "Plans for discourse". *Intentions for Communication*, pp 417-444, MIT Press.

Gruber T. R. (1993) "A translation approach to portable ontology specification". *Knowledge Acquisition*, 5(2):199-220, 1993

Haddadi A. (1996) "Communication and cooperation in agent systems. A pragmatic theory". En *Lectures Notes in Artificial Intelligence 1056*. Springer-Verlag, 45-49. 1996.

Hayes-Roth B. (1985) "A blackboard architecture for control". *Artificial Intelligence*, 26(3):251-321.

Hoare C. A. R. (1978) "Communicating sequential processes". *Communications of the ACM*, 21:666-677.

Labrou Y. y Finin T. (1994) "A semantics approach for KQML – a general purpose communication language for software agents". En *Third International Conference on Information and Knowledge Management*, Noviembre 1994.

Labrou Y. y Finin T. (1997) "Comments on the specification for FIPA '97 Agent Communication Language*". University of Maryland, Baltimore County*, Baltimore Maryland USA, February 28, 1997.

Labrou Y. y Finin T. (1997) "Semantics for an Agent Communication Language". *Departamento de Ciencias Computacionales e Ingeniería Eléctrica. Universidad de Maryland Baltimore County*, Baltimore, Maryland 21250 USA.

Labrou Y., y Finin T. (1997) "A proposal for a new KQML Specification". *Reporte técnico. Departamento de Ciencias Computacionales e Ingeniería Eléctrica. Universidad de Maryland Baltimore County*, Baltimore, Maryland 21250 USA. TR CS-97-03. Febrero, 1997.

Mayfield J., Labrou Y. y Finin T. (1995) "Desiderata for agent communication language". En *Proceedings de 1995 AAAI Spring Symposium on Information Gathering in Distributed Environments*, Marzo 1995.

Mayfield J., Labrou Y. y Finin T. (1996) "Evaluation of KQML as an agent communication language". En *Intelligent Agents Volume II – Proceedings of the 1995 Workshop on Agent Theories. Architectures, and Languages*. M. Wooldridge, J.P. Muller and M. Tambe (eds). Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996.

Mayfield J., Labrou Y. y Finnin T. (1995), 'Evaluation of KQML as an Agent Communication Language', *INTELLIGENT AGENTS II: Agent Theories, Architectures and Languages*, pp 347-361, Wooldridge, Michael J, Mueller, P and Tambe, Milind, (Eds), Springer Verlag, Berlin 1995.

Russell S. y Norvig P. (1995) *Artificial Intelligence: A Modern Approach*. Englewood

Cliffs, NJ: Prentice-Hall, 1995.

Schelling T. C. (1960) *The Strategy of Conflict*. Harvard University Press, Cambridge, MA.

Searle J. R. (1969) *Speech Acts*. Cambridge University Press.

Searle J. R. (1979) *Expression and Meaning*. Cambridge University Press.

Sperber D., Wilson D. (1986) *Relevance: Communication and Cognition*, Blackwell.

Vanderveken D. (1992) *Meaning and Speech Acts*. Cambridge University Press.

Werner E. (1989) "Cooperating agents: A unified theory of communication and social structure". *Distributed Artificial Volumen II*, pp 3-36. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA.

White J. E. (1994) "Telescript techonology: The foundation for the electronic marketplace". *White paper, General Magic*, Inc., 2465 Latham Street, Mountain View, CA 94049, 1994.

Winograd T., Flores F. (1986) *Understanding Computers and Cognition: A new Foundation for Design*. Ablex Publishing Corp.